ARMY RESEARCH LABORATORY

# Large-Scale Integrated Process Modeling Simulations Enabling Composite Materiel Developments and Applications

**by Brian J. Henz and Dale R. Shires**

**ARL-TR-3680** **December 2005**

## NOTICES

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

**Army Research Laboratory**
Aberdeen Proving Ground, MD 21005-5067

**ARL-TR-3680** **December 2005**

# Large-Scale Integrated Process Modeling Simulations Enabling Composite Materiel Developments and Applications

**Brian J. Henz and Dale R. Shires**
**Computational and Information Sciences Directorate, ARL**

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED (From - To) |
|---|---|---|
| December 2005 | Final | 2002–2004 |

**4. TITLE AND SUBTITLE**
Large-Scale Integrated Process Modeling Simulations Enabling Composite Materiel Developments and Applications

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**
Brian J. Henz and Dale R. Shires

**5d. PROJECT NUMBER**
5U03CC

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-CI-HC
Aberdeen Proving Ground, MD 21005-5067

**8. PERFORMING ORGANIZATION REPORT NUMBER**
ARL-TR-3680

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
High Performance Computing Modernization Program Office
1010 North Glebe Road, Suite 510
Arlington, VA 22201

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
A virtual manufacturing environment that provides modules for predicting the process-induced residual stresses in polymeric composite materials has recently gone through extensive testing by the authors. The use of polymeric composite materials in Department of Defense materiel developments has made it increasingly important to predict the service life and the mechanical responses of such structures. Process-induced behavior plays a critical role in the accurate modeling of mechanical responses. Predicting process-induced residual stresses of composite material structures requires the coupling of resin infusion, heat transfer, and multiscale thermal residual stress models. The complexity of modeling the process-induced effects requires the use of modern software engineering techniques with multiphysics coupled models. The model and software developmental efforts are described in this report.

**15. SUBJECT TERMS**
resin transfer molding, finite-element method, parallel, OOP, beta test

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Brian J. Henz |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL | 32 | 19b. TELEPHONE NUMBER (*Include area code*) 410-278-6531 |

## Contents

## List of Figures



## List of Tables

## Acknowledgments

This work was supported in part by a grant of computer time and resources by the Department of Defense High Performance Computing Modernization Program Office. Additional support was provided by the U.S. Army Research Laboratory Major Shared Resource Center.

INTENTIONALLY LEFT BLANK.

# 1. Introduction

Accurate analysis of the mechanical response of composite structural components requires that process-induced residual stresses be known. These process-induced residual stresses can be computed by using multidisciplinary manufacturing process models. Modeling of complex manufacturing processes under the liquid composite molding family, namely resin transfer molding (RTM), requires the integration of multidisciplinary numerical analysis tools of the physical models. These tools can be combined in many ways including common data file formats or development in a common programming framework. For legacy codes that have been developed, optimized, and debugged over time, it may be more economical to implement a common data file format such as the eXtensible Data Model and Format (XDMF) from the U.S. Army Research Laboratory (ARL) (*1*, *2*). For new software tools being developed or parallelized, the authors have created the Simple Parallel Object-Oriented Computing Environment for the finite-element method (SPOOCEFEM) (*3*, *4*). The SPOOCEFEM framework is used exclusively for development of the engineering applications in this report.

# 2. The SPOOCEFEM Framework

The SPOOCEFEM framework was developed expressly for the purpose of reducing the amount of time required for scientific software development, to promote code growth and maturation, and to improve code coupling capabilities. The library avoids being overly generic but successfully compartmentalizes many of the functions common to the FEM approach. It has been developed in the C++ language and has been ported to numerous computing platforms including Intel-based Linux clusters, the SGI Origin, and the IBM Power series machines. While the initial ramp-up time to develop the fully functioning SPOOCEFEM library was significant, the library is already being successfully used in several codes.

SPOOCEFEM uses numerous technologies to provide a complete development environment. At the lower levels of the framework are high level language compilers for C++, C, and Fortran. On top of these compilers are the typically vendor supplied OpenGL visualization libraries, the Message-Passing Interface (MPI) library for parallel interprocessor communication, and linear algebra packages such as BLAS and LAPACK. Next are the open source packages such as the visualization packages found in the Visualization Toolkit, binary and text file storage utilities, and linear system solver libraries. SPOOCEFEM sits atop all of this, along with specialized matrix and vector templates as shown in figure 1, so that the low level library calls are obscured from the application developer.

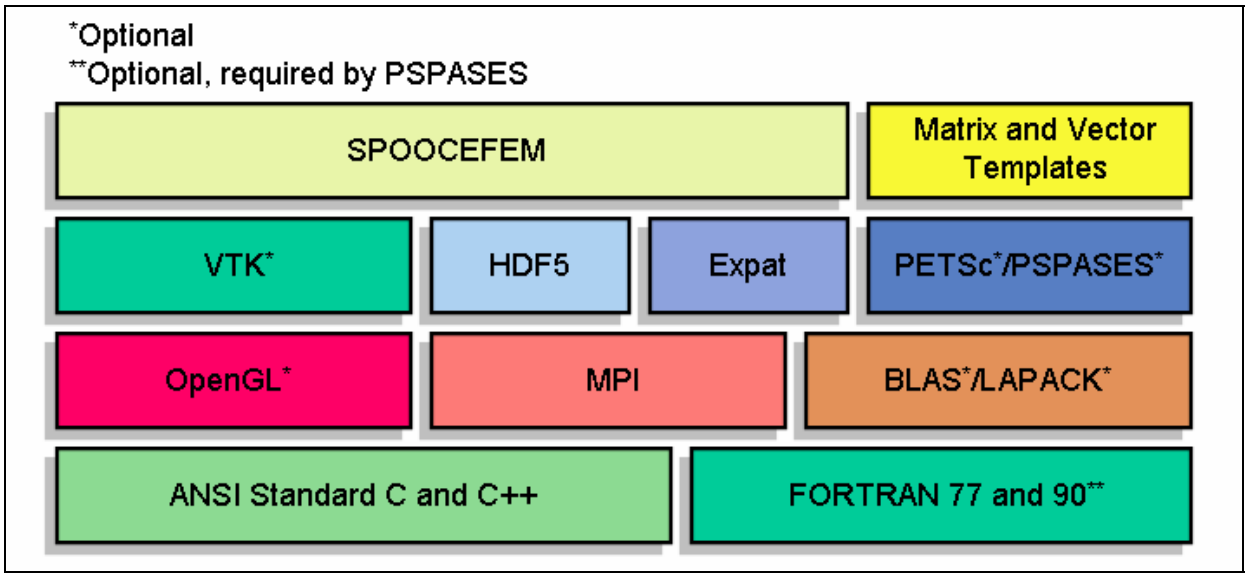



Figure 1. SPOOCFEM building block framework.

SPOOCEFEM utilizes many of today's newest technologies to provide a complete integrated data file format. It uses XDMF for data storage (*1*). This data file format is extensible and designed for storing data used and generated by numerical applications. SPOOCEFEM only utilizes a subset of the larger XDMF structure for unstructured grid problems as encountered in the FEM. Data is segregated into descriptive text data (stored in XML) and large binary data sets (stored in the Hierarchical Data Format). Using XDMF guarantees the ability to quickly use visualization tools already available such as Paraview from Kitware (www.kitware.com). More details on SPOOCEFEM are outside the scope of this report but are referenced elsewhere (*3*, *4*).

## 3. Modeling the RTM Process

Modeling of the RTM process includes the analysis of underlying physical phenomena of fluid flow, heat transfer, and induced material stresses. Each of these models will now be briefly described, and then the integration within the SPOOCEFEM framework will be discussed.

### 3.1 Resin Flow in Porous Media

Initially, the RTM process may be modeled as isothermal flow of resin through a porous media, namely a fibrous preform. An implicit algorithm for this purpose based on the Pure FEM was developed by Mohan et al. (*5*) and Ngo et al. (*6*). This algorithm was then implemented and parallelized in three incarnations. The first was a High Performance Fortran version utilizing

data parallelism. The second version was developed with Fortran 90 and MPI for message-passing parallelism (*7*, *8*). The current version utilizes SPOOCEFEM and is also based on message-passing parallelism, but with many of the bookkeeping routines managed by the SPOOCEFEM libraries (*4*). This isothermal resin flow modeling application is called the Composite Manufacturing and Process Simulation Environment (COMPOSE).

The transient mass conservation of the infusing resin is considered with a state variable defining the infused state of a node. Application of the Galerkin weighted residual to the physical model equation leads to a semidiscrete equation of the form shown in equation 1. Further details can be found in Mohan et al. (*5*).

$$C\dot{\Psi} + KP = q\,, \tag{1}$$

where

$$C = \int_{\Omega} N^T N d\Omega\ , \tag{2}$$

$$K = \int_{\Omega} B^T \frac{\overline{K}}{\mu} B d\Omega\,, \tag{3}$$

$$q = \int_{\Gamma} N^T \left( \frac{\overline{K}}{\mu} \cdot \nabla P \cdot n \right) d\Gamma\ , \tag{4}$$

and

$$\dot{\Psi} = \frac{\Psi_{n+1} - \Psi_n}{\Delta t}\ . \tag{5}$$

$\Psi$ is the nodal fill fraction, $\mathbf{0} \le \Psi \le \mathbf{1}$, defining the infused state of a physical node location, $\overline{K}$ is the permeability tensor of the fibrous preform, and $P$ is the pressure measured at a node. The boundary conditions for equation 1 are given as follows:

$$\frac{\partial P}{\partial n} = \mathbf{0} \text{ on mold walls,} \tag{6}$$

$$P = \mathbf{0} \text{ at flow front,} \tag{7}$$

and

$$P = P_{\mathbf{0}} \text{ prescribed pressure at inlet} \tag{8}$$

or

$$q = q_0 \text{ prescribed flow rate at inlet,} \tag{9}$$

where $P_0$ and $q_0$ represent prescribed pressure and flow rate at the inlet(s), respectively. Initially, at time t = 0,

$$\Psi = \mathbf{1} \text{ at the inlet} \tag{10}$$

and

$$\Psi = \mathbf{0} \text{ elsewhere.} \tag{11}$$

In equation 2, $N$ is the elemental shape function, and in equation 3, $B = \nabla N$.

### 3.1.1 Software Development

Utilizing the object-oriented design (OOD) of SPOOCEFEM the FEM problem is divided into classes. Most of the computational work besides the linear equation solvers in COMPOSE takes place in the various element classes. A diagram showing the `ComposeElement2dTri3` class hierarchy is shown in figure 2. COMPOSE is capable of modeling the RTM process with 2.5-D or full 3-D analysis. For composite components typically seen in the aerospace industry, the 2.5-D model is most heavily utilized. The 2.5-D element types in COMPOSE are three-noded triangles and four-noded quadrilaterals. Multiple inheritance is used in the COMPOSE element classes in order to make code development more efficient. The `ComposeElement` class provides functionality common to all COMPOSE element types such as storage of viscosity data. The `ComposeElement2d` class provides mass matrix and local permeability computations. The functionality unique for each COMPOSE element type such as stiffness matrix computation is available in the `ComposeElement2dTri3` class. More details on the development of the SPOOCEFEM-based COMPOSE software are available in Henz and Shires (*9*).

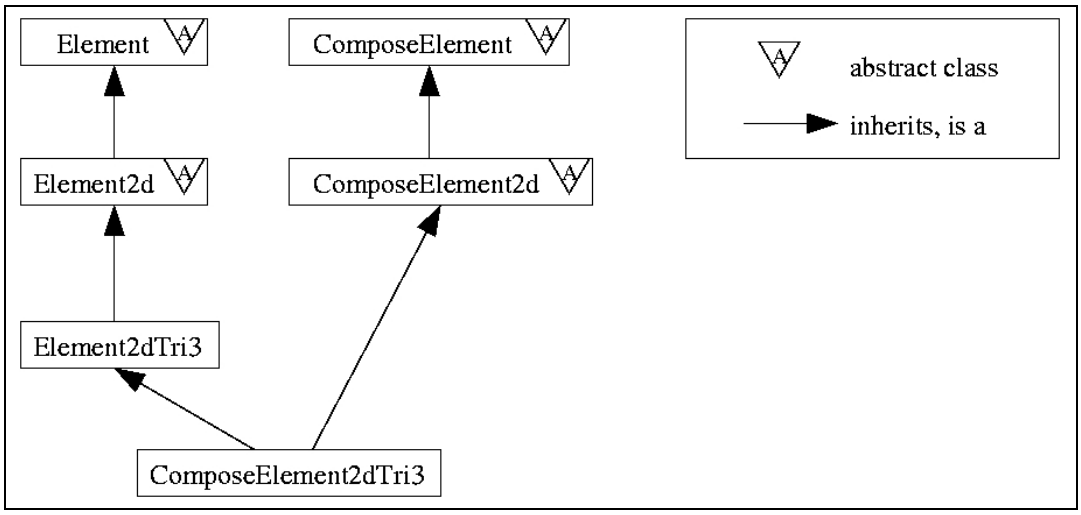



Figure 2. Hierarchy of the `ComposeElement2dTri3` class showing multiple inheritance.

Development time has decreased with the utilization of OOD from 12 months for the Fortran 90 version of COMPOSE to ~3 months for the C++ version (*10*). In addition, code reuse of the OOD library is highlighted in the remainder of this report. Use of OOD in scientific computing always brings with it performance concerns associated with the runtime system (*11*, *12*). Our own experience shows that OOD does not necessarily degrade performance. The COMPOSE software that was originally developed in FORTRAN 77 was re-implemented with OOD in C++. Table 1 shows a break out of the three most time-consuming routines in COMPOSE, the first of which is the linear solver with ~96% of the total runtime. The faster execution of the C++ version of the solver is most likely due to small enhancements performed during translation. The other routines (calculate pressure and update) are noticeably slower in C++ than FORTRAN. The three fold increase in runtime for the update routine is due to the numerous accesses to values contained in the vector templates. The advantage of these templates is that they allow quick use of numerous solvers. Therefore, this overhead is not critical in many instances and considered a good trade-off with the increased software development productivity.

Table 1. Function-oriented and OOD runtime comparison (in seconds).

| Operation | FORTRAN 77 | C++ |
|---|---|---|
| Linear solver | 2766 | 2694 |
| Calculate pressure | 114 | 128 |
| Update | 22 | 71 |
| Core total time | 2902 | 2893 |

### 3.1.2 Test Results

The isothermal resin flow model COMPOSE was comprehensively tested in October of 2001. This original parallel code used Fortran 90 with MPI for parallelism. The COMPOSE software was tested for scalability, validity, and the results were verified across various numbers of processors. The performance of the Fortran 90 version of COMPOSE can be seen in figure 3.

The COMPOSE verification and validation included comparison with analytic and experimental results. Figure 4 shows the analytic model geometry and some numerical results. Figure 5 shows the experimental comparison.

Verification of COMPOSE was performed using the data in figure 4 where results from various processor counts are compared for consistency.

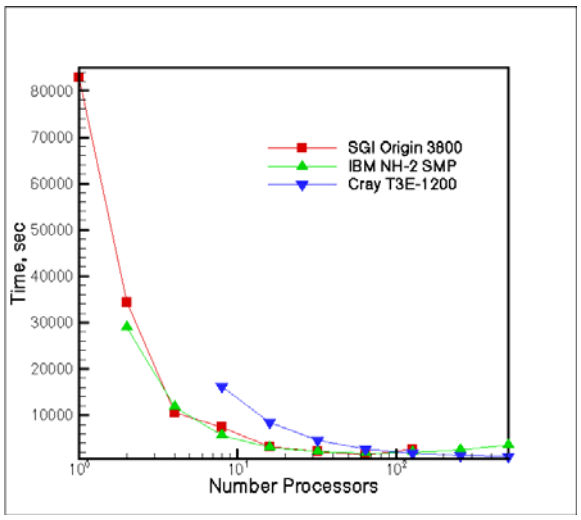



Figure 3a: COMPOSE model with 135,492 DOFs.

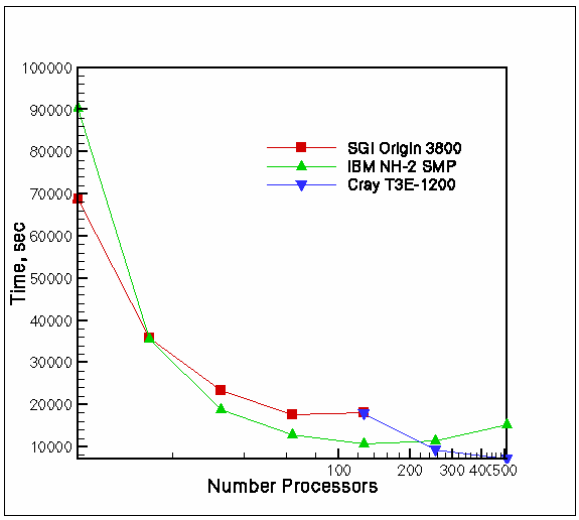



Figure 3b: COMPOSE model with 405,327 DOFs.

Figure 3. Wall clock time required to solve two COMPOSE models on three different architectures.

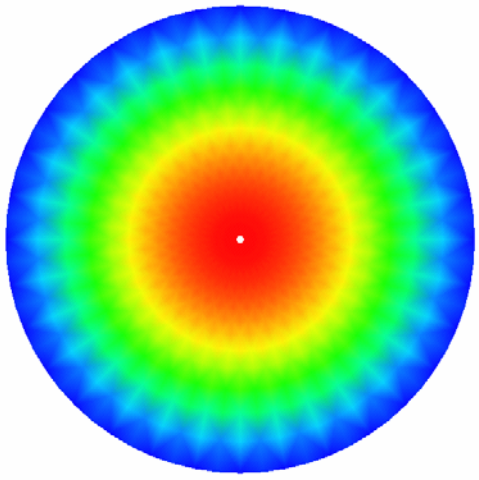


Figure 4a: Geometry of thin disk that has analytic solutions for flow front and injection pressure.

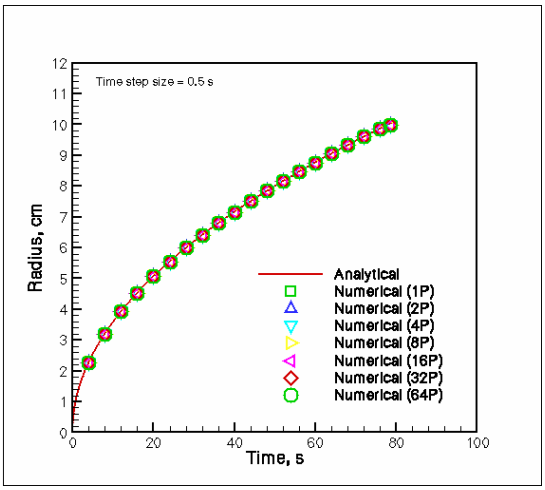



Figure 4b: Radial flow front position comparing analytic and numerical results.

Figure 4. Experimental validation results for Fortran 90 version of COMPOSE.

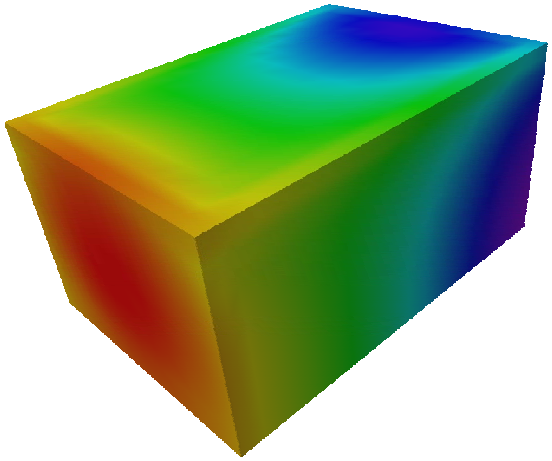


Figure 5a: Computational model of geometry used in experimental validation of COMPOSE.

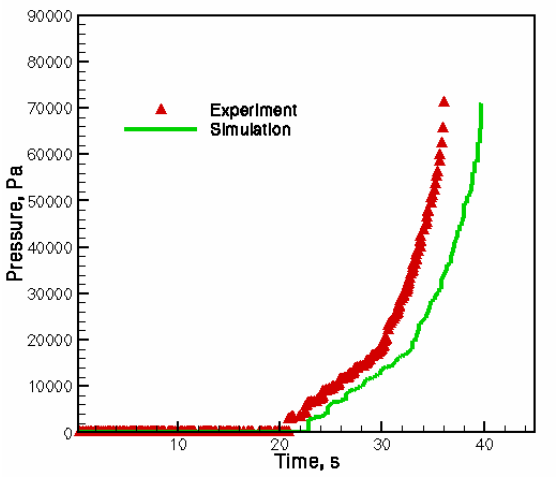



Figure 5b: Pressure sensor data from computational model and experiment showing excellent agreement.

Figure 5. Validation results for Fortran 90 version of COMPOSE using experimental data.

### 3.2 Convection, Conduction, and the Exothermic Resin Curing Process

For the RTM process, it is often only necessary to model the isothermal resin flow to analyze the process accurately. In some cases, though, the heat transfer and resin cure kinetics must also be considered in order to understand the manufacturing process to allow for the analysis and computation of the thermally induced residual stresses. Extending the COMPOSE RTM model to include heat transfer and resin cure kinetics resulted in the multidisciplinary software called PhoenixFlow. PhoenixFlow utilizes the resin flow model developed for COMPOSE with inheritance and OOD.

#### 3.2.1 Heat Transfer

The resin flow modeled in COMPOSE and PhoenixFlow is based on an integral form of the mass balance equation coupled with Darcy's law by way of a pure finite-element-based methodology (*5, 13*). The thermal model equations employ an energy balance equation based on a fiber-resin thermal equilibrium model (*14*). Utilizing the FEM in order to discretize the problem domain for the heat transfer model in PhoenixFlow is given as

$$C\dot{T} + \left(K_{ad} + K_{cond}\right)T = Q_q + Q_{\dot{G}}, \tag{12}$$

where

$$C = \int_{\Omega} W^T \rho c_p N d\Omega , \tag{13}$$

$$K_{ad} = \int_{\Omega} W^T \rho c_p \left(u \cdot B_N\right) B d\Omega , \tag{14}$$

$$K_{cond} = \int_{\Omega} B_W^T k B_N d\Omega , \tag{15}$$

$$Q_q = \int_{\Gamma} W^T \left(-q \cdot n\right) d\Gamma , \tag{16}$$

and

$$Q_{\dot{G}} = \int_{\Gamma} W^T \phi \dot{G} d\Gamma . \tag{17}$$

$B_W$ is the spatial derivative of the weighting function, $W$, and $B_N$ is the spatial derivative of the finite-element shape function, $N$. $\phi$ is porosity, $\rho$ is density, $c_p$ is specific heat, $k$ is the thermal conductivity, $u$ is the resin velocity, and $\dot{G}$ is the rate of heat generation by chemical reactions or other sources. The subscripts $f$ and $r$ denote fiber and resin properties, respectively. The boundary conditions required to solve equation 5 are given by

$$T = T_W \text{ at the mold wall,} \tag{18}$$

$$T = T_{r\mathbf{0}} \text{ at the mold inlet,} \tag{19}$$

$$k \frac{\partial T}{\partial n} = \left(\mathbf{1} - \phi\right) \rho_r c_{pr} u \cdot n \left(T_{f\mathbf{0}} - T\right) \text{ at the resin front,} \tag{20}$$

and

$$T\left(t = \mathbf{0}\right) = T_W \text{ initially.} \tag{21}$$

### 3.2.2 Resin Cure Model

In addition to the temperature calculations, the degree of resin cure is also computed in the current developments. The discretized system of equations is given as

$$C\dot{\alpha} + K\alpha = Q_{R_\alpha} , \tag{22}$$

where

$$C = \int_{\Omega} \phi W^T N d\Omega, \tag{23}$$

$$K = \int_{\Omega} \phi W^T \left(u \cdot B_N\right) d\Omega, \tag{24}$$

and

$$Q_{R_\alpha} = \int_{\Gamma} \phi W^T R_\alpha d\Omega. \tag{25}$$

$B_N$ in equation 24 is the spatial derivative of the finite element shape function, $N$. $\alpha$ is the degree of cure, and $R_\alpha$ is the rate of chemical reaction. The boundary conditions for equation 22 are given as

$$\alpha = \mathbf{0} \text{ at the mold inlet} \tag{26}$$

and

$$\alpha\left(t = \mathbf{0}\right) = \mathbf{0} \text{ initially.} \tag{27}$$

The resin curing process is exothermic. The heat generated, $\dot{G}$, is given by

$$\dot{G} = H_R R_\alpha, \tag{28}$$

where $H_R$ is the heat of reaction per unit volume for the pure resin (*15–17*).

### 3.2.3 Software Development

The first step in extending COMPOSE for nonisothermal analysis is to develop a new element type that includes routines for computing and storing stiffness matrices for the additional thermal models. Graphically, this extension is shown in figure 6. The `PhoenixFlowElement` class contains routines required by all PhoenixFlow element types. These routines include calculating effective material properties and the elemental viscosities. The `PhoenixFlowElement2d` class does not contain any functionality at this time but is used for convenience in the class hierarchy. The `PhoenixFlowElement2dTri3` class provides functionality for computing elemental stiffness matrices for heat transfer and resin cure.

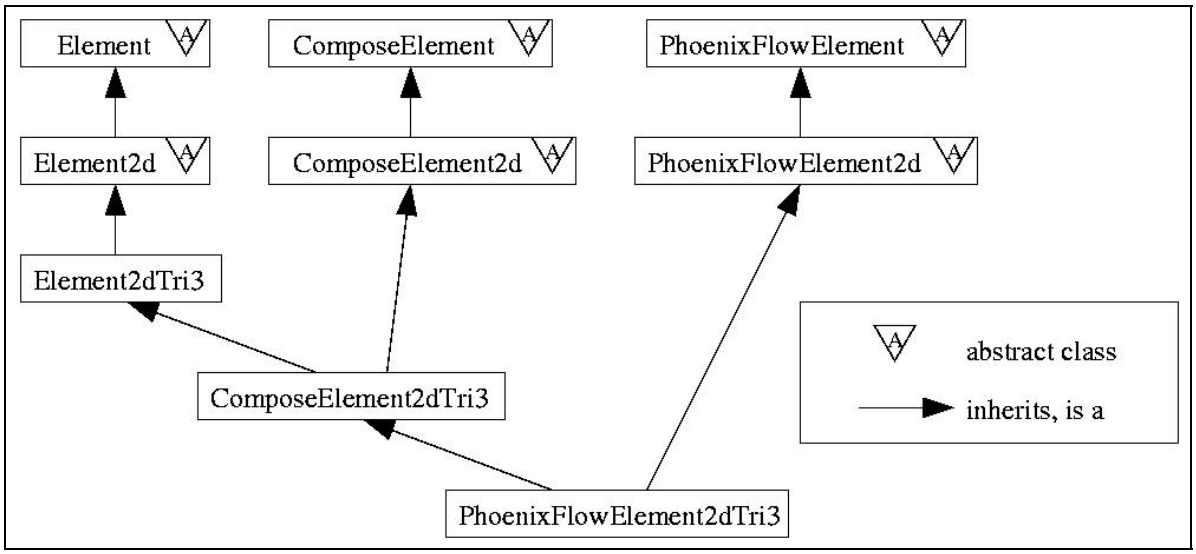



Figure 6. Hierarchy of the `PhoenixFlowElement2dTri3` class showing multiple inheritance.

### 3.2.4 Test Results

The PhoenixFlow application was developed as part of the Common High Performance Computing Software Support Initiative System of Systems (SOS) portfolio. This project, managed by the Department of Defense High Performance Computing Modernization Program, requires in-depth testing for correctness, scalability, and utility. Like COMPOSE, PhoenixFlow was tested against these hallmarks. All of these metrics were defined in the test plan as Critical Technical Parameters (CTPs) with the scalability CTP given as an example in table 2.

It is interesting to note that PhoenixFlow has superlinear speedup for all architectures on some number of processors and for some architectures on any number of processors. The poor scalability on the IBM machine for greater than 32 processors can be explained by the communications system employed on that architecture. On the IBM, the processors are grouped onto nodes with 32 processors and shared memory on the node. Between nodes, the memory is not shared and the communication system between nodes is much slower than the on-node system. This communication architecture manifests itself as the poor scalability results shown in figure 7. In addition to scalability, PhoenixFlow has been validated with published experimental results and subsequently verified across architectures and numbers of processors. Figure 8 contains the verification results for one experiment, three architectures, and one element type. The full verification suite includes three experiments, three architectures, and two element types, not to mention cross-architecture comparisons that were performed.

The scalability results for the PhoenixFlow software are given in figure 7.

Table 2. Scalability CTP from SOS test report given as an example.

| CTP Title | Parameters | | Worst Case Tested Value and Outcome |
|---|---|---|---|
| | **Optimum Objectives** | **Minimum Threshold** | |
| Scalable software suites:<br><br>Demonstrate reduction in clock time as a function of increased central processing units | Fixed speedup exceeds 60% of optimum on 64 processors | Fixed speedup exceeds 50% of optimum on 32 processors | Tested Value:<br>• SGI O3K: 120.75%<br>• Linux Cluster: 111.30%<br>• IBM SP4: 64.73%<br><br>Outcome:<br>___ Baseline established<br>___ Fails to meet minimum threshold<br>___ Meets minimum threshold<br>_✓_ Meets optimum objective |
| Discussion:<br><br>As was stated in the Beta Test Plan, the timing data here do not include the time required to compute the flow or compute post-filling cure. Rather, it involves only the time to compute the thermal solution.<br><br>These results were computed using relative speedup which is defined as:<br>$\hat{S}_p = \frac{p_{\min} T_{p_{\min}}}{T_p}$,<br>where $p_{\min}$ is the smallest number of processors that were used to compute a solution and $T$ is the time associated with the given processor count. The minimum processor count used on all architectures in this case was eight. This seems to represent a good choice to try and mitigate possible hierarchical memory effects and also to limit job waiting as these trials were conducted in a production environment. | | | |

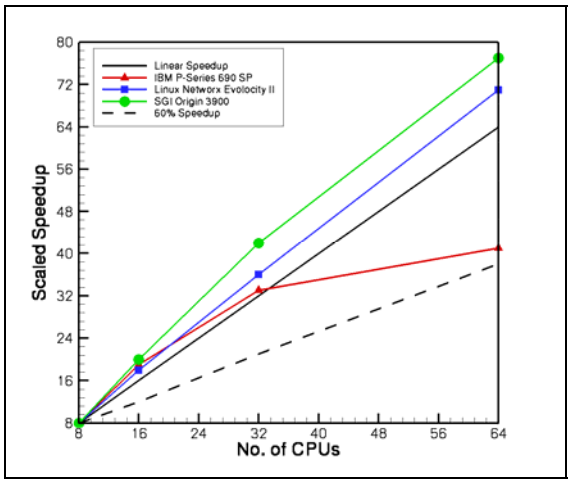



Figure 7. Scalability of PhoenixFlow software on various architectures.

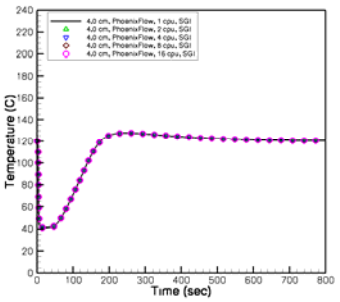



Figure 8a: SGI, location 1.

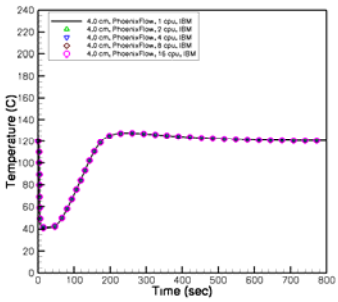



Figure 8b: IBM, location 1.

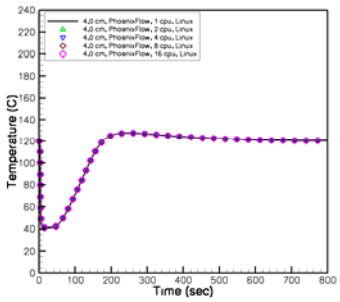



Figure 8c: Linux Cluster, location 3.

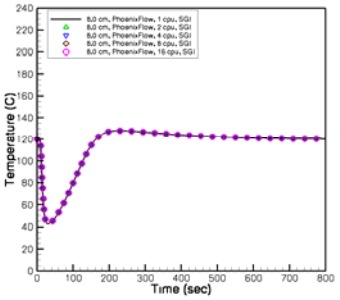



Figure 8d: SGI, location 2.

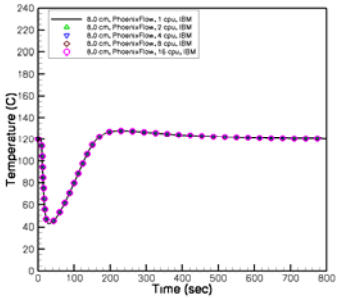



Figure 8e: IBM, location 2.

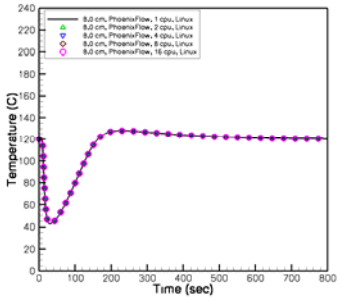



Figure 8f: Linux Cluster, location 2

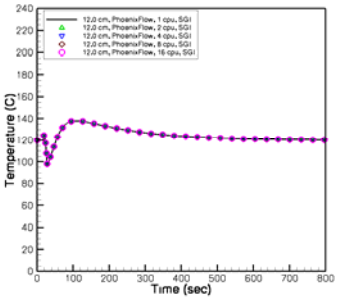



Figure 8g: SGI, location 3.

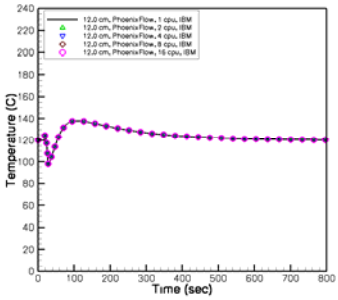



Figure 8h: IBM, location 3.

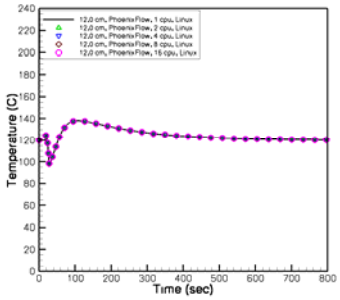



Figure 8i: Linux Cluster, location 3.

Figure 8. Verification results for PhoenixFlow on three architecture and 1, 2, 4, 8, and 16 processors.

The validation of PhoenixFlow involved comparing numerical results with published experimental results. These comparisons are shown in figure 9 for three parallel architectures.

The plots in figure 9 show that the PhoenixFlow model is accurate and consistently predicts nonisothermal RTM results in real world experiments. The PhoenixFlow application was also validated against three published geometries, three transient sets of data per publication, and three architectures.

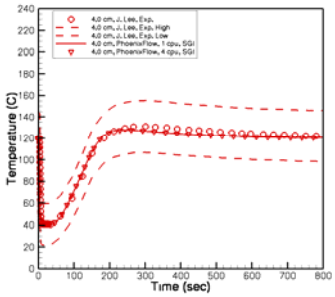



Figure 9a: SGI, location 1.

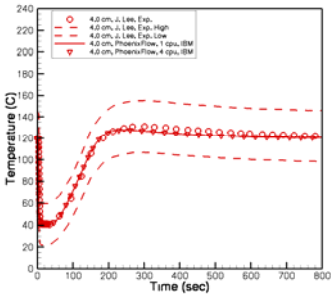



Figure 9b: IBM, location 1.

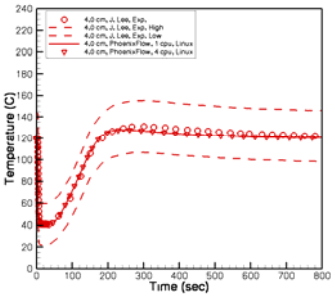



Figure 9c: Linux Cluster, location 3.

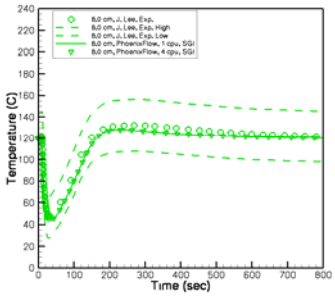



Figure 9d: SGI, location 2.

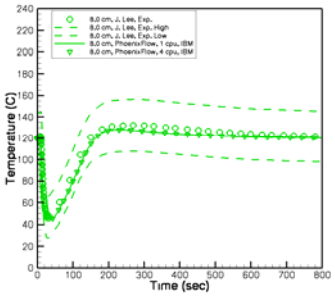



Figure 9e: IBM, location 2.

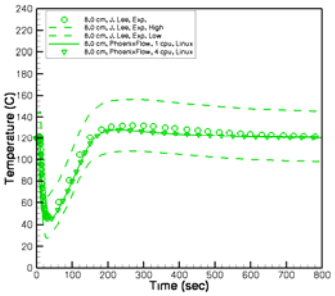



Figure 9f: Linux Cluster, location 2.

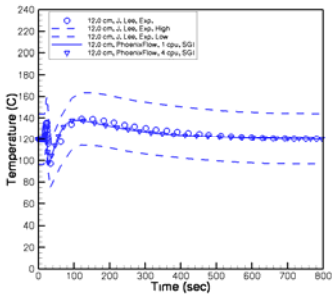



Figure 9g: SGI, location 3.

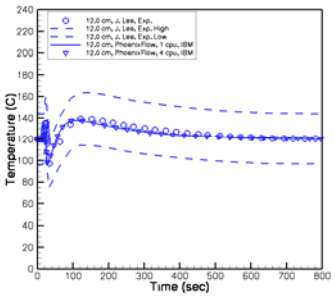



Figure 9h: IBM, location 3.

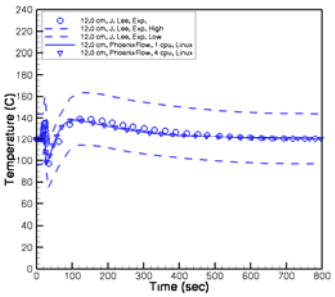



Figure 9i: Linux Cluster, location 3.

Figure 9. Validation results of PhoenixFlow for experiment 1, showing window of valid results around published results.

### 3.3 Multiscale Residual Thermal Stress Analysis

Composite components manufactured with processes such as RTM often contain residual stresses after processing. These residual stresses are typically produced during the cooldown phase after the resin has cured. The model used to compute residual stresses in the current developments is based on the Multiscale Elastic Thermal Stress software developed by Chung et al. (*18*). The model also predicts homogeneous material properties for heterogeneous materials through the Asymptotic Expansion Homogenization method (*18*).

The thermal residual stresses are computed from the following equation:

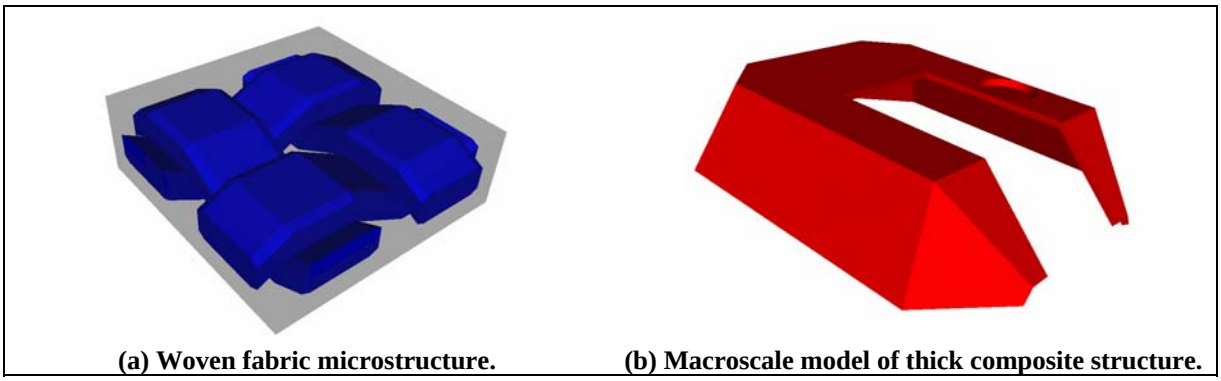
$$\left[\int_{\Omega} B^T DBd\Omega\right]\Delta u_{n+1} = \int_{\Omega} B^T D\varepsilon^{th}_{n+1} d\Omega + \int_{\Gamma} N^T \tau_{n+1} d\Gamma + \int_{\Omega} N^T F_{n+1} d\Omega - \int_{\Omega} B^T \sigma_n d\Omega\,, \qquad (29)$$

where $\sigma$ is the stress experienced by the body, $F$ is the body force per unit volume, $B$ is the derivative of the shape function $N$, $\tau = \sigma \cdot n$, $u$ is the nodal displacement, and $\varepsilon$ is the elemental strain. A detailed derivation of the preceding equation can be found in Ngo (*14*).

The software to compute the residual thermal stresses developed at ARL utilizes a baseline academic code from Ngo (*14*). The original code was developed using FORTRAN 77, was serial only, and did not contain a common data file format or output other than simple elemental results in text files. A restructured, enhanced, and parallelized application was developed in SPOOCEFEM to provide the same capabilities. The new SPOOCEFEM-based software developed at ARL is called MSStress (MultiScale thermal residual Stress analysis).

### 3.3.1 Software Development

Since the MSStress software is currently coupled with COMPOSE and PhoenixFlow through common data files (XDMF), no inheritance of classes from previous applications is used. The key portion of the software development for MSStress is the implementation within the SPOOCEFEM framework so that it can be parallelized and have a graphical user interface (GUI) available for pre- and postprocessing capabilities. Some interesting details of the software development for MSStress are provided. The microscale model in MSStress is a unit cell of the composite component, as shown in figure 10a, that is used to compute the homogenized material properties for the macroscale model, shown in figure 10b. As was soon discovered, the microscale model was much more difficult to parallelize because of the symmetric boundary conditions required in the numerical analysis. After considering this for some time, we concluded that the microscale model was typically somewhat smaller than the macroscale model and could therefore be computed in serial on each processor used in the parallel analysis. The macroscale problem is subsequently solved in parallel with the computed material properties. The adverse affect on performance from not parallelizing the microscale model is insufficient to offset the extra effort that would be required to parallelize the symmetric boundary condition routines at this time. A more detailed analysis of the MSStress software development is not in the scope of this report and will appear in future work.



**(a) Woven fabric microstructure.** **(b) Macroscale model of thick composite structure.**

Figure 10. Visual example of micro and macroscale MSStress models.

### 3.3.2 Test Results

The MSStress software was not subject to the rigorous testing requirements of the other components since it was not part of the original code development plan. MSStress was only required to output data in a common format useable by a third-party application for analysis in a continuum mechanics code. In this case, we used P-DINOSAURUS (*19*) as the continuum mechanics application for service condition loading. Some of these results can be seen in figure 11.

---

## 4. Computational Model Integration

---

This section discusses of some of the issues encountered while developing tightly coupled codes (such as COMPOSE and PhoenixFlow) and when loosely coupling codes through data I/O only. The first section discusses the tight coupling of the flow model in COMPOSE with the thermal and cure models in PhoenixFlow. The second section provides details about using XDMF in order to couple the thermal results from PhoenixFlow with the multiscale residual stress model in MSStress.

### 4.1 Tightly Coupled Flow/Temperature/Cure

In modeling the RTM process, the fluid flow and thermal models must be tightly coupled. This coupling is accomplished through software engineering. The COMPOSE fluid flow model is implemented in the SPOOCEFEM framework (*9*) as a stand-alone code. The thermal and cure models in PhoenixFlow then take advantage of the COMPOSE software through inheritance and OOD.

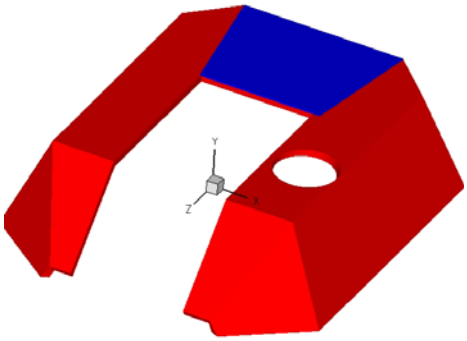


Figure 11a: Representative vehicle geometry showing region of blast loading.

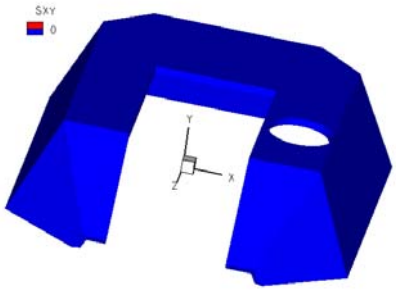



Figure 11b: Representative part with no residual stress information.

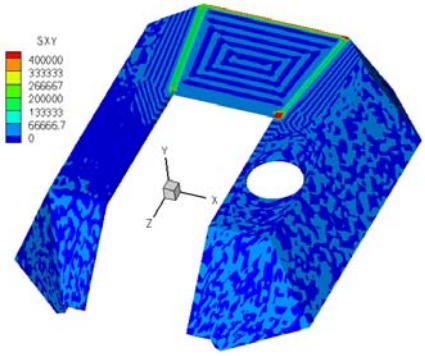



Figure 11c: Shear stress results from P-DINOSAURUS at the 20th time step (with no process induced residual stress).

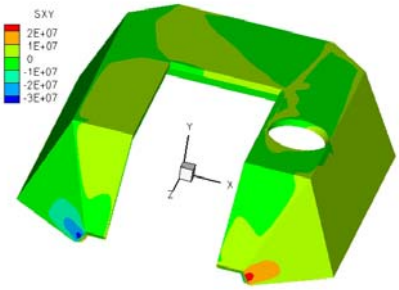



Figure 11d: Initial shear stress computed by MSStress.

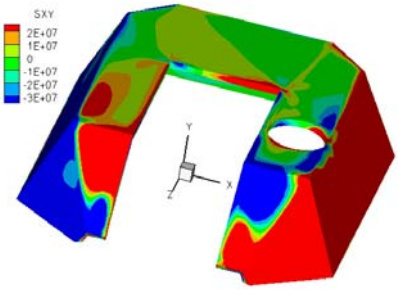



Figure 11e: P-DINOSAURUS results of same blast loading using the process induced initial residual stresses in the material.

Figure 11. Thick geometry showing process induced residual stress effects on sample service loading conditions.

An example of this OOD is the three-noded triangular element in PhoenixFlow. The COMPOSE three-noded triangular element has storage, computation, and assembly routines for the fluid flow stiffness matrix. The `PhoenixFlowElement2dTri3` class inherits these routines and adds similar routines for thermal and cure analysis. In COMPOSE, the viscosity of the resin inside an element remains constant, whereas the resin viscosity is computed at each flow time step in PhoenixFlow. For the storage of the resin viscosity the `ComposeElement` class is

utilized, and new routines for resin viscosity computation are added to the `PhoenixFlowElement` class.

In addition to the element classes, the `PhoenixFlowFEM` class inherits functionality from the `ComposeFEM` class. This inherited functionality includes the one-shot filling routine (*20*) and sensitivity analysis for temperature independent parameters (*21*). The main loop in the COMPOSE software computes filling at each time step until the mold is filled. Since temperature and degree of cure must be computed in between filling time steps, a new routine in PhoenixFlow has been developed that computes a single filling time step and then proceeds to thermal and cure analysis before continuing the resin flow process.

Another issue in the thermal analysis includes heat transfer in the through-thickness direction causing high temperature gradients. This requires that 3-D elements be built up on top of the 2-D flow elements, as seen in figure 12. These extra elements and nodes increase the complexity and storage requirements of the linear equations many-fold in PhoenixFlow. As such, careful attention is paid to the nodal numbering and elemental connectivity in order to improve performance and reduce code complexity.

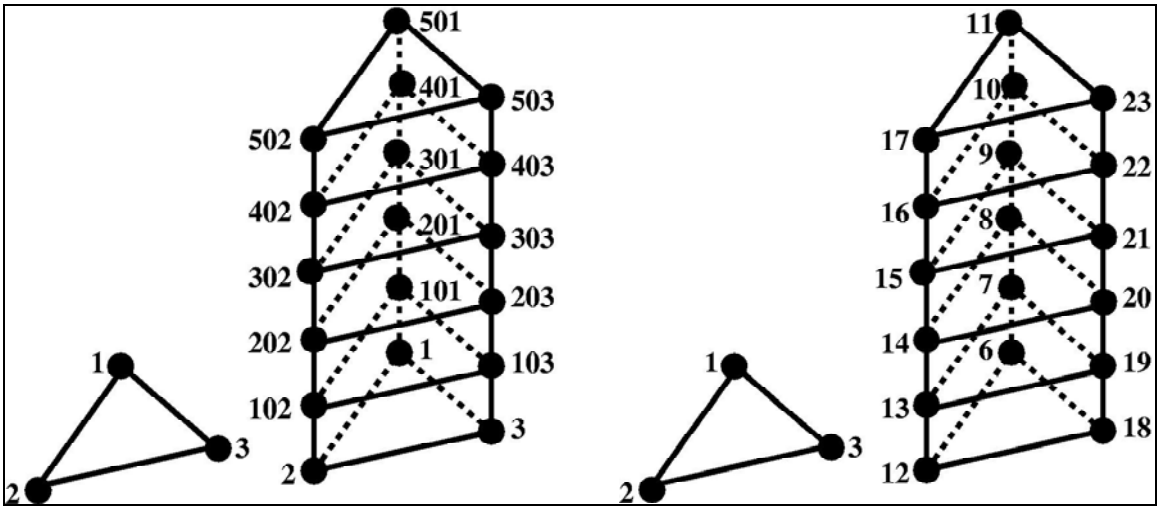
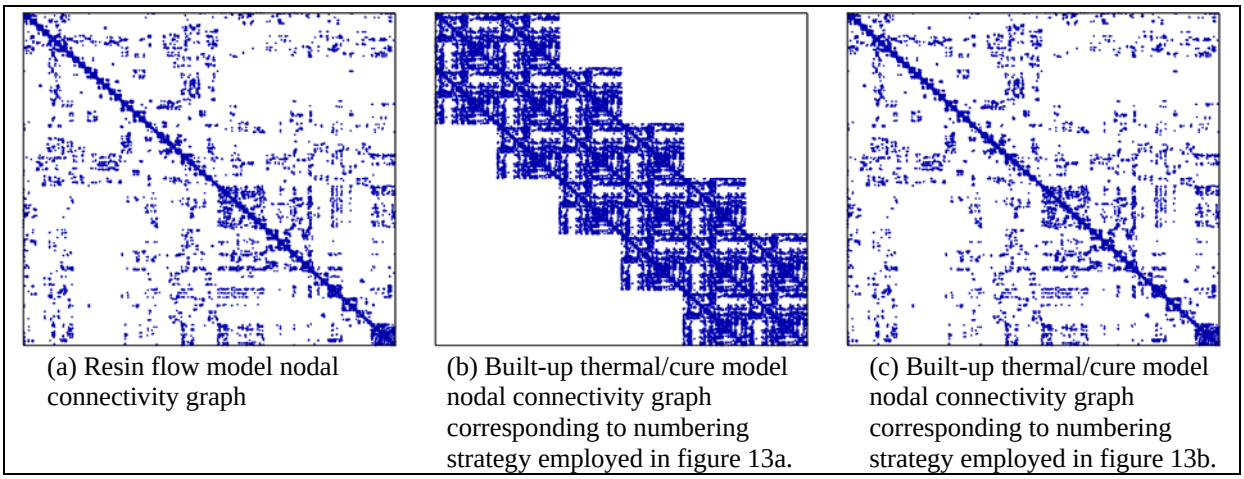



Figure 12. Stack of six-noded wedge elements for PhoenixFlow thermal/cure analysis.

The importance of the nodal numbering strategy shown in figure 12 is evident in the connectivity graphs shown in figure 13. Figure 13a shows the nodal connectivity graph of a certain problem for the resin flow model. In other words, this graph shows how the various nodes in the finite-element mesh are connected by the elements. Locality of reference is an important consideration in today's parallel supercomputers as the memory systems used are highly complex and layered. Minimization of the connectivity graph bandwidth can lead to improvements in the runtime and cache efficiency of a code using these data structures (*22*). A Reverse Cuthill-McKee (RCM) (*23*) pass can be used to optimize the locality and produce a tight diagonal for the connectivity



(a) Resin flow model nodal connectivity graph

(b) Built-up thermal/cure model nodal connectivity graph corresponding to numbering strategy employed in figure 13a.

(c) Built-up thermal/cure model nodal connectivity graph corresponding to numbering strategy employed in figure 13b.

Figure 13. Comparison of previous connectivity graphs for resin flow vs. thermal/cure models.

graphs. An RCM pass performed on the connectivity graph of the resin flow model (figure 13a) will be maintained for the thermal/cure model (figure 13c) since they are of similar structure.

### 4.2 Coupled Flow/Thermal/Cure/Stress Analysis

The thermal residual stress analysis software (MSStress) requires temperature profiles from the nonisothermal RTM software (PhoenixFlow). The integration of the MSStress software with PhoenixFlow is accomplished by storing data in the XDMF file format. This format standardizes the storage and description of data so that MSStress can read temperature results from PhoenixFlow.

There are two ways in which the residual stresses can be computed in the RTM process. In the first, the stresses are computed incrementally at each thermal time step as the resin cures, which would require tight coupling of the PhoenixFlow and MSStress codes. In the second method, the residual stresses are assumed to be insignificant prior to completion of the curing process, at which time residual stresses begin to develop from cooling of the completely cured composite component. MSStress computes residual stresses using the second method, thus allowing for this loose coupling.

Currently, MSStress only supports eight-noded hexahedral elements. This requires that the thermal analysis be performed with 8-noded hexahedral, 6-noded wedge, or 4-noded tetrahedral elements in a full 3-D analysis or with quadrilateral elements that are built-up for thermal analysis as described earlier. This method is often preferable since many composite structures in the aerospace industry are thin. In order to accurately and efficiently model the flow and thermal problems, a 2.5-D method is employed. Full 3-D models are available in COMPOSE and PhoenixFlow, but these often require large finite element models compared to an equivalent 2.5-D model when applicable.

Even though software tools were previously available to model the flow/thermal/cure mold filling process and to compute the subsequent thermal residual stresses, these codes were immature and resided only in the academic world. They lacked optimization, GUIs, standardized I/O, and parallelism. All of these beneficial components are part of the SPOOCEFEM framework and hence are now part of the COMPOSE, PhoenixFlow, and MSStress codes. The basic functionality of these codes has also been easily increased beyond the initial capabilities of the academic codes due to the constructive nature of the SPOOCEFEM framework. We feel these developments represent a compelling case for the use of OOD in multidisciplinary computational software.

## 5. Conclusions

As demonstrated here, SPOOCEFEM provides a framework for the development of coupled multidisciplinary computational codes. In conjunction with XDMF, SPOOCEFEM facilitates the coupling of developmental and legacy scientific codes. This coupling of codes is required in order to model complex processes such as the RTM process. In addition, the software developed with SPOOCEFEM can be easily parallelized and ported in order to take advantage of high-performance computers including clusters of personal computers. The combination of COMPOSE/PhoenixFlow/MSStress models the RTM manufacturing process from resin injection through curing and finally outputs process-induced residual stresses in a composite component. The residual stresses can then used by codes such as DYNA3D (*24*) or P-DINOSAURUS (*19*) as an initial material stress configuration for subsequent dynamic loading analysis. This coupling process is becoming more important as multiscale and multiphysics problems are analyzed.

## 6. References


1. Clarke, J. A.; Namburu; R. R. A Distributed Commuting Environment for Interdisciplinary Applications. *Concurrency and Computation: Practice and Experience* **2002**, *14*, 1–14.

2. Clarke, J. A.; Schmitt, C. E.; Hare, J. J. Developing a Full Featured Application From an Existing Code Using the Distributed Interactive Computing Environment. *DOD High Performance Computing Modernization Program Users Group Conference*, Houston, TX, 1998.

3. Henz, B. J.; Shires, D. R.; Mohan, R. V. A Composite Manufacturing Process Simulation Environment (COMPOSE) Utilizing Parallel Processing and Object-Oriented Techniques. *International Conference on Parallel Processing*, Vancouver, B.C., Canada, 2002.

4. Shires, D. R.; Henz, B. J. An Object-Oriented Approach for Parallel Finite Element Analysis. *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, 2003.

5. Mohan, R. V.; Ngo, N. D.; Tamma, K. K. On a Pure Finite Element Methodology for Resin Transfer Mold Filling Simulations. *Polymer Engineering and Science* **1999**, *39*, (1), 26–43.

6. Ngo, N. D.; Mohan, R. V.; Chung, P. W.; Tamma, K. K. Recent Developments Encompassing NonIsothermal/Isothermal Liquid Composite Molding Process Modeling/Analysis: Physically Accurate, Computationally Effective and Affordable Simulations and Validations. *Journal of Thermoplastic Composite Materials* **1998**, *11* (6), 493–532.

7. Shires, D. R.; Mohan, R. V.; Mark, A. Optimization and Performance of a Fortran 90 MPIBased Unstructured Code on Large Scale Parallel Systems. *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, June 2001.

8. Shires, D. R.; Mohan, R. V.; Mark, A. An Evaluation of HPF and MPI Approaches and Peformance in Unstructured Finite Element Simulations. *Journal of Mathematical Modeling and Algorithms* **2002**, *1*, 153–167.

9. Henz, B. J.; Shires, D. R. Development and Performance Analysis of a Parallel Finite Element Application Implemented in an Object-Oriented Programming Framework. *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, June 2003.

10. Shires, D. R.; Henz, B. J. Lessons Learned and Perspectives on Successful HPC Software Engineering and Development. *International Conference on Software Engineering Research and Practice*, Las Vegas, NV, June 2004.

11. Post, D.; Kendall, R. *Software Project Management and Quality Engineering Practices for Complex, Coupled Multi-Physics Massively Parallel Computational Simulations: Lessons Learned from ASCI*; LA-UR-03-1274; Los Alamos Laboratory: Los Almos, NM, 2003.

12. Veldhuizen, T. L.; Jurnigan, M. E. Will C++ be Faster than FORTRAN? *First International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE)*, 1997.

13. Mohan, R. V.; Shires, D. R.; Mark, A.; Tamma, K. K. Advanced Manufacturing of Large Scale Composite Structures: Process Modeling, Manufacturing Simulations and Massively Parallel Computing Platforms. *Journal of Advances in Engineering Software* **1998**, *29* (3–6), 249–264.

14. Ngo, N. D. Computational Developments for Simulation Based Design: Multi-Disciplinary Flow/Thermal/Cure/Stress Modeling, Analysis, and Validation for Processing of Composites. Ph.D. Thesis, University of Minnesota, MN, 2001.

15. Sourour, S.; Kamal, M. R. SPE technical paper, *18* (93), 1972.

16. Kamal, M. R.; Sourour, S. Integrated Thermo-Rheological Analysis of the Cure of Thermosets. SPE technical paper, *18* (187), 1973.

17. Kamal, M. R.; Sourour, S. SPE technical paper, *13* (59), 1973.

18. Chung, P. W.; Tamma, K. K.; Namburu, R. R. Asymptotic Expansion Homogenization for Heterogeneous Media: Computational Issues and Applications. *Composites - Part A: Applied Science and Manufacturing* **2001**, *32* (9), 1291–1301.

19. Kanapady, R; Tamma, K. K. *P-DINOSAURUS: Parallel Dynamic INtegration Operators for Structural Analysis Using Robust Unified Schemes*; Department of Mechanical Engineering: University of Minnesota, MN, 2002.

20. Voller, V. R.; Chen, Y. F. Prediction of Filling Times of Porous Cavities. *International Journal for Numerical Methods in Fluids* **1996**, *23*, 661–672.

21. Henz, B. J.; Tamma, K. K.; Kanapady, R.; Ngo, N. D.; Chung, P. W. *Process Modeling of Composites by Resin Transfer Molding: Sensitivity Analysis for Isothermal Considerations*. AIAA-2002-0790, 40th Aerospace Sciences Meeting, Reno, NV, January 2002.

22. Kumfert, G.; Pothen, A. *Two Improved Algorithms for Envelope and Wavefront Reduction*. BIT, 1997, *37* (3), 559–590.

23. Cuthill, E.; McKee, J. Reducing the Bandwidth of Sparse Symmetric Matrices. *Proceedings of the 24th National Conference, Association for Computing Machinery*, 1969, pp 157–172.

24. Lin, J. I. *DYNA3D: A Nonlinear, Explicit, Three-Dimensional Finite Element Code for Solid and Structural Mechanics*; Lawrence Livermore National Laboratory: Oak Ridge, TN, 1998.

NO. OF
COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD
SYSTEMS OF SYSTEMS INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN
AUSTIN TX 78759-5316

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE ALC IMS
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 1 | HIGH PERFORMANCE COMPUTING MODERNIZATION PROG OFFICE<br>A MARK<br>1010 GLEBE RD STE 510<br>ARLINGTON VA 22201 |